AFRL-IF-RS-TM-2003-1
In-House Technical Memorandum
February 2003

GRAPHICAL INTERFACE CONCEPT FOR A SIGNAL DETECTION PROCESS

Brian Costello

AIR FORCE RESEARCH LABORATORY
INFORMATION DIRECTORATE
ROME RESEARCH SITE
ROME, NEW YORK

AFRL-IF-RS-TM-2003-1 has been reviewed and is approved for publication.

APPROVED: [illegible]

GERALD C. NETHERCOTT
Chief, Multi-Sensor Exploitation Branch
Information and Intelligence Exploitation Division

FOR THE DIRECTOR: [illegible]

JOSEPH CAMERA
Chief, Information & Intelligence Exploitation Division
Information Directorate

REPORT DOCUMENTATION PAGE	*Form Approved OMB No. 074-0188*

1. AGENCY USE ONLY (Leave blank)	**2. REPORT DATE** FEBRUARY 2003	**3. REPORT TYPE AND DATES COVERED** **In House Tech Memo, May 2002 – August 2002**

4. TITLE AND SUBTITLE GRAPHICAL INTERFACE CONCEPT FOR A SIGNAL DETECTION PROCESS	**5. FUNDING NUMBERS** PE - 62702F PR - 459E TA - PR WU - OJ
6. AUTHOR(S) Brian Costello	
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) AFRL/IFEC 32 Brooks Road Rome, NY 13441-4114	**8. PERFORMING ORGANIZATION REPORT NUMBER** AFRL-IF-RS-TM-2003-1
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) AFRL/IFEC 32 Brooks Road Rome, NY 13441-4114	**10. SPONSORING / MONITORING AGENCY REPORT NUMBER** AFRL-IF-RS-TM-2003-1

11. SUPPLEMENTARY NOTES
AFRL Project Engineer: Dr. Andrew Noga/IFEC/315-330-2270/Andrew.noga@rl.af.mil
Brian Costello is a participant in AFRL's summer student engineering employment program.

12a. DISTRIBUTION / AVAILABILITY STATEMENT APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.	**12b. DISTRIBUTION CODE**

13. ABSTRACT ***(Maximum 200 Words)***

A concept is presented for a graphical interface to the Adjustable Bandwidth Concept (ABC) signal energy detection process, U.S. Patent 5,257,211. To verify the utility and effectiveness of the interface, a MATLAB (Mathworks, Inc) implementation has been developed. The emphasis is on those applications which would require real-time processing. Although the implementation itself does not run in real-time for higher sampling rates, it does serve as a design that can be used in the development of such systems.

14. SUBJECT TERMS war simulation, computer games graphical user interface, signal detection, signal grouping	**15. NUMBER OF PAGES** 36
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT UNCLASSIFIED	**18. SECURITY CLASSIFICATION OF THIS PAGE** UNCLASSIFIED	**19. SECURITY CLASSIFICATION OF ABSTRACT** UNCLASSIFIED	**20. LIMITATION OF ABSTRACT** UL

Table of Contents

1.0 Introduction

This report covers work done under the Summer Engineering Aide Program, for the year 2002. Work was concentrated in the building of a prototype graphical user interface for a real-time implementation of the ABC Process.

1.1 Overview of the ABC Process

The ABC Process is a signal detection algorithm that uses various stages of time and frequency averaging. This algorithm was developed within AFRL/IFEC, and is covered under U.S. Patent #5,257,211.

The ABC algorithm incrementally increases time averaging while decreasing frequency averaging. By using multiple stages of this averaging, the ABC process is able to detect signals with a wide range of bandwidths. This allows the operator to see all of the signals in the region of interest, even if they are overlapped.

1.2 Background

During the 1999 Summer Engineering Aide program, a Matlab based program was developed that ran the ABC algorithm on a user-defined .wav file. This work was documented in [1]. This program is satisfactory for off-line applications, however, there is also interest in a real-time implementation of the ABC process. The issue then arose that the software engineers developing the new implementation would not know what parameters the RF engineers would want to be able to control. The solution was to create a prototype implementation that would simulate running at real-time, and would show the developers what the engineers and operators would like to see and be able to control. This prototype implementation is discussed herein.

2.0 Prototype GUI Implementation

The main goal of this implementation was not necessarily to run at real-time, but more to emulate running at real-time. To do this, existing functions were used, and some new ones were created. The existing ABC program was written in Matlab; therefore, Matlab was chosen to be the basis for this implementation. Another existing program, a running spectrogram, was also written in Matlab and facilitated further development.

The new GUI is made up of three windows: the ABC-Analyzer window, the Signal Object Display, and a waitbar.

2.1 ABC-Analyzer Window

The ABC-Analyzer Window is derived from an existing program, fft_scan, that was written by AFRL/IFEC Dr. Andrew Noga. As seen in Figures 2.1.1 and 2.1.2, it shows the input spectrogram as well as spectrograms for all of the ABC stage outputs. It also contains menus to edit all of the relevant ABC parameters.

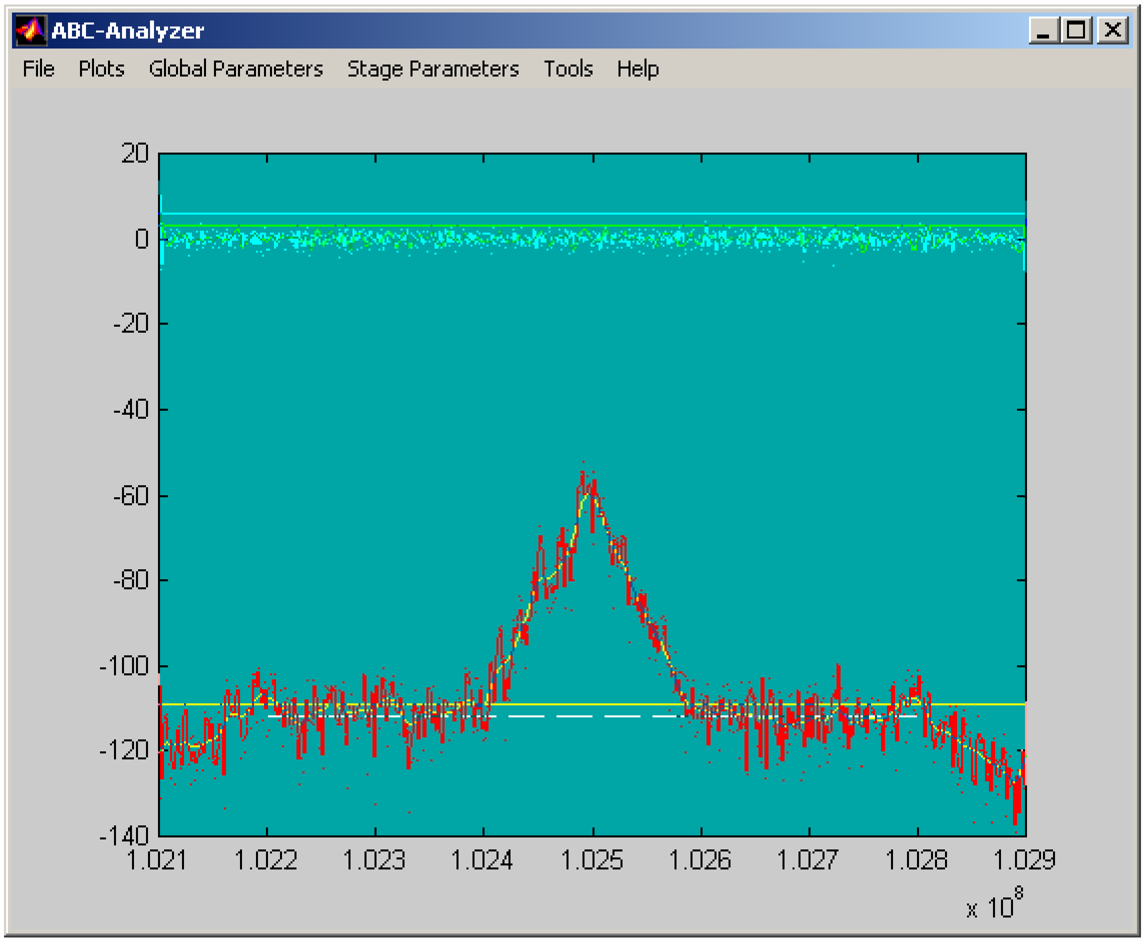


Fig 2.1.1 Screen Shot of the ABC-Analyzer Window

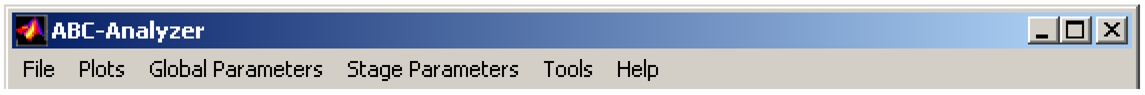


Fig 2.1.2 The ABC-Analyzer Menu Bar

2.1.1 ABC-Analyzer Menus

Referring to Figure 2.1.3, the first menu in the ABC-Analyzer window is the 'File' menu It contains general options that apply to either the file as a whole, or to the running of the ABC program. This includes information related to the type of filtering. It also contains options to save the current display, and to exit the program. The filter information is used for the running calculation of the noise floor estimate. An option to change the running average factor was also included in an effort to give the operator the maximum control possible.

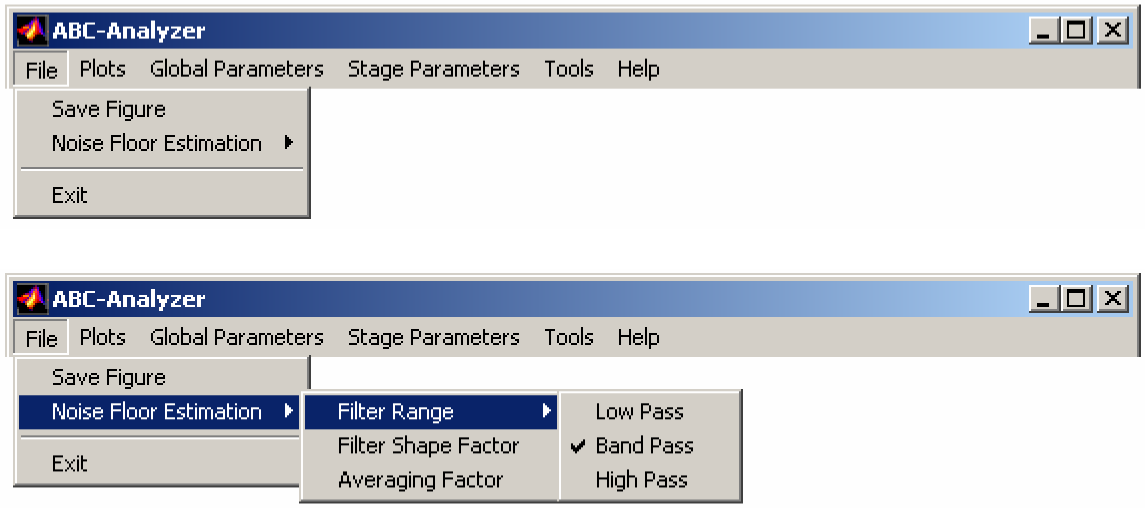


Fig 2.1.3 ABC-Analyzer 'File' Menu and Sub-Menus

.

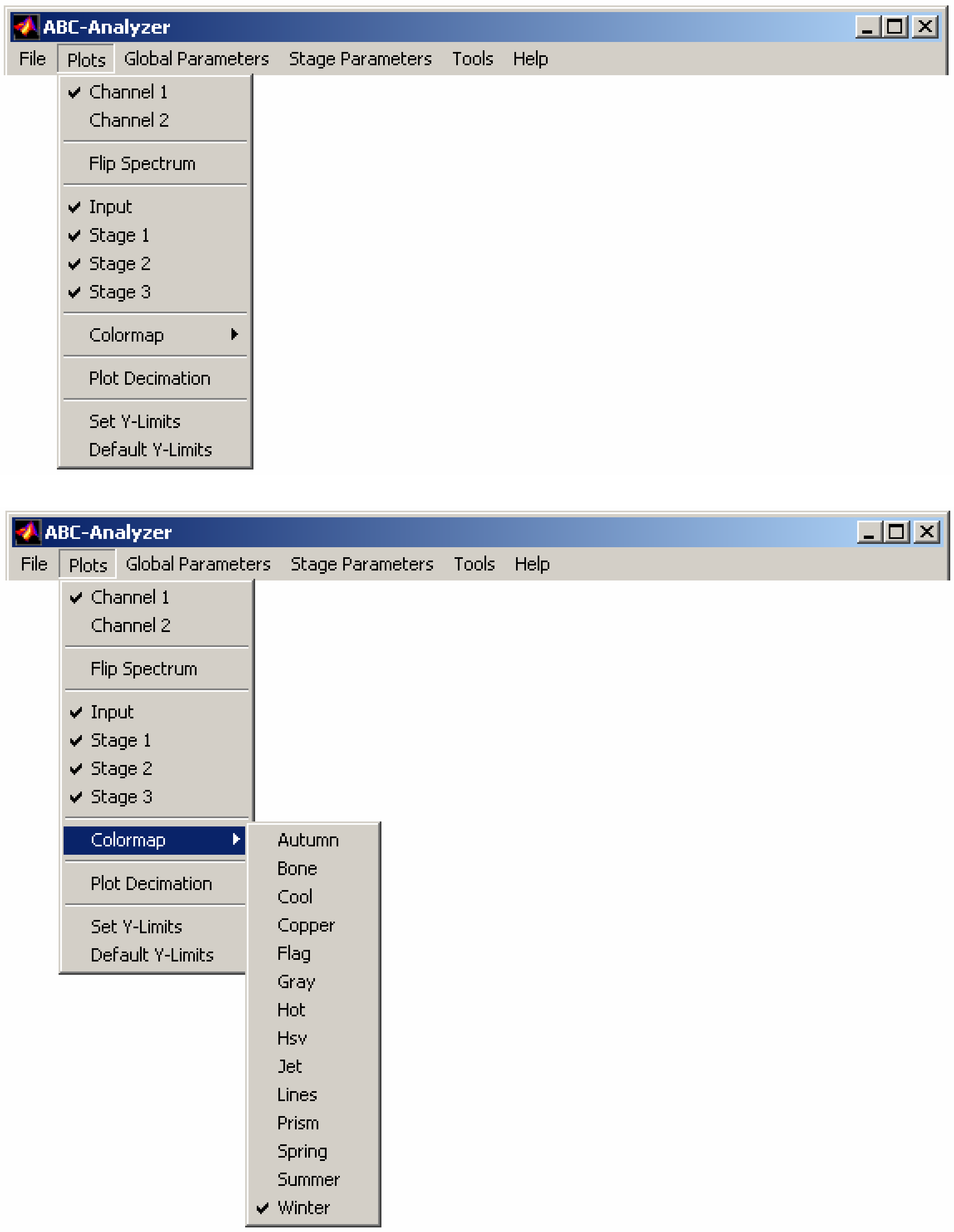


Fig 2.1.4 ABC-Analyzer ‘Plots’ Menu and Sub-Menu

Referring to Figure 2.1.4, the ‘Plots’ menu is the second menu in the window. This menu allows the user to adapt the display to their current needs and preferences. Options include the ability to switch between channels (if the data has multiple channels), flip the spectrum left to right, turn on and off the different stage plots as desired, change the color scheme, the ability to change the refresh rate of the display, and to change the range of the vertical axes. It should be noted that the ABC algorithm is set to calculate on the selected channel only. Also, turning off the plot of one stage does not stop that stage from being calculated. It only stops that stage from being seen on the display. Likewise, turning a stage off in the ABC-Analyzer window does not turn it off in the Signal Object Display. The 'Flip Spectrum' option works similarly in that it only flips the spectrum visually in the ABC-Analyzer window. Any output will be calculated as if the flip option was turned off. Next, the 'colormap' option allows the operator to change the color scheme that is used for the different plots. The submenu for this option contains all of the colormaps that come standard in Matlab. The 'Plot Decimation' option changes the number of time segments between refreshes of the display. However, every time segment is still calculated and run through the algorithm regardless of the decimation setting. Unlike the last option though, the plot decimation does apply to both display windows. Finally, The Y-Limit options allow the user to edit the range for the vertical axis. The 'Default Y-Limits' sets the range from –100 dBx to 20 dBx.

Referring to Figure 2.1.5, the 'Global Parameters' menu allows the user to edit ABC parameters that apply to the entire algorithm. It contains the options to change the FFT size, to allow a 50% FFT overlap, and to change the number of stages. The ‘FFT Size’ submenu contains options that are powers of two, ranging from 256 frequency bins to 8192 bins. The ‘FFT Overlap?’ option toggles the use of a 50% FFT overlap in the running of the ABC algorithm.

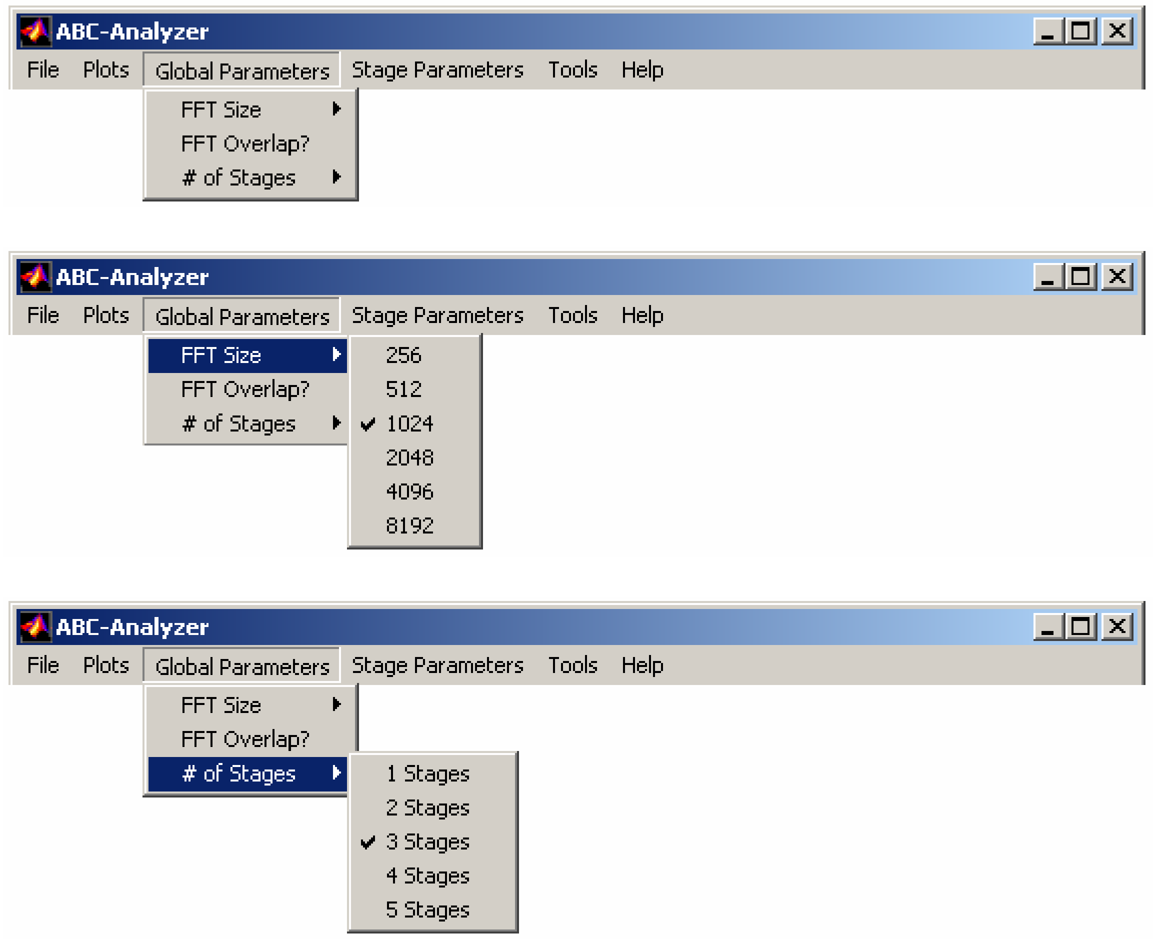


Fig 2.1.5 ABC-Analyzer ‘Global Parameters’ Menu and Sub-Menus

The ‘# of Stages’ submenu allows the user to change the number of ABC processing stages that the algorithm is using. The options for this menu range from one to five, as a number higher than five seemed to increase processing time without any significant increase in the detection capability.

The next menu is the 'Stage Parameters' menu. (See Figures 2.1.6 through 2.1.8) This menu allows the operator to adjust ABC parameters that affect each stage of the algorithm individually. The submenus for this option differ depending on which stage is selected.

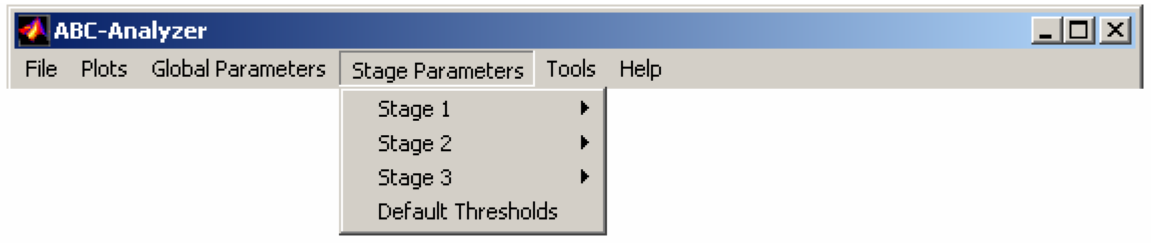


Fig 2.1.6 ABC-Analyzer 'Stage Parameters' Menu

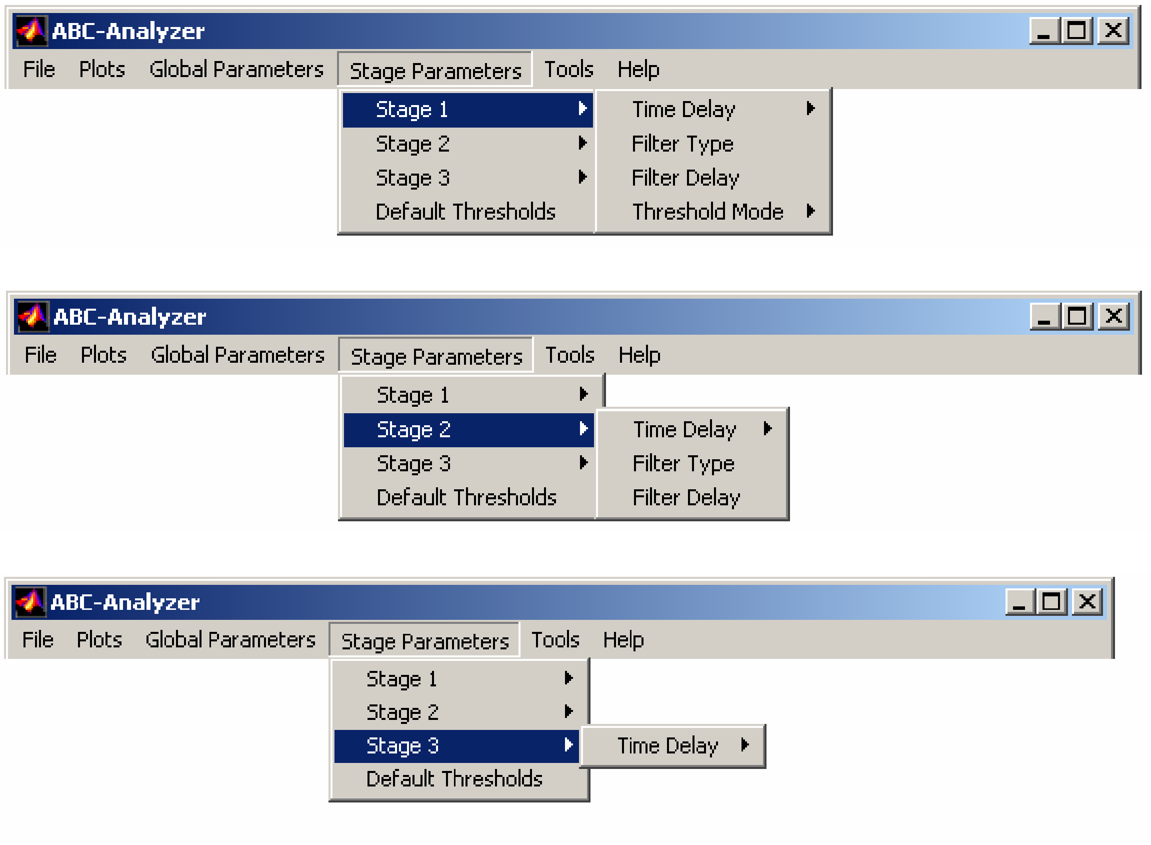


Fig 2.1.7 ABC-Analyzer 'Stage Parameters' Sub-Menus

The ‘Time Delay’ option is used to adjust the amount of time averaging, and has possible values from zero to nineteen. This option can be found in all of the stage submenus.

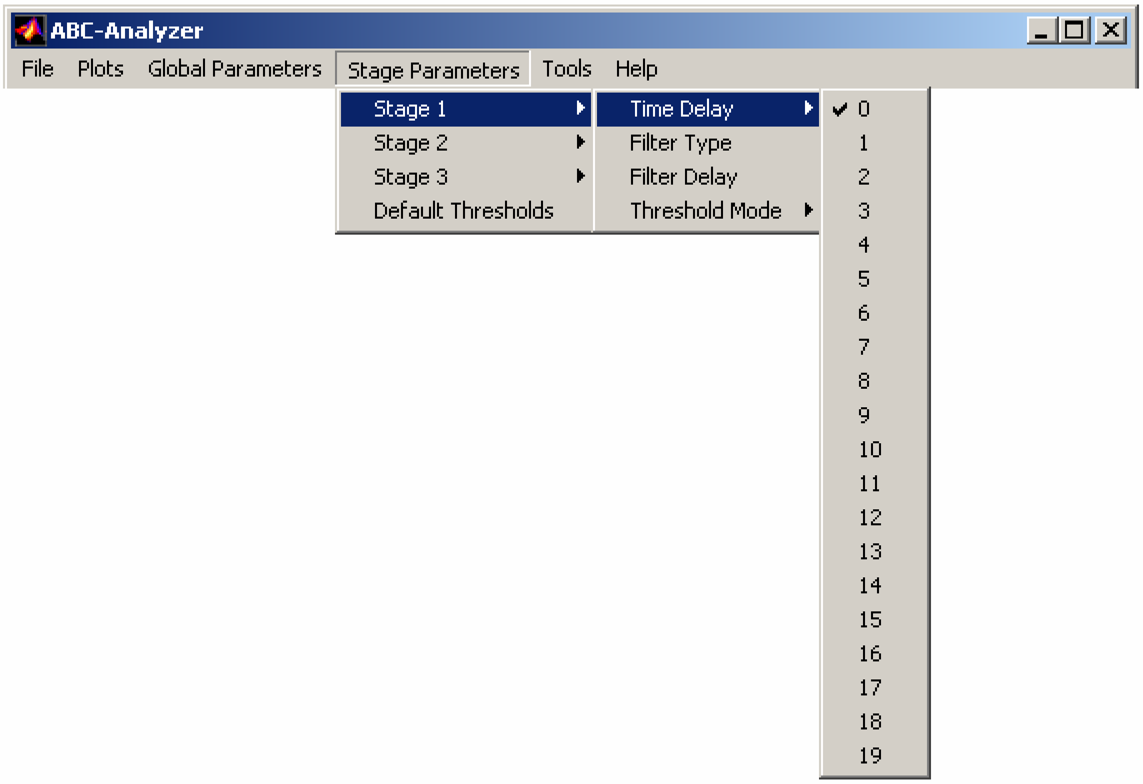
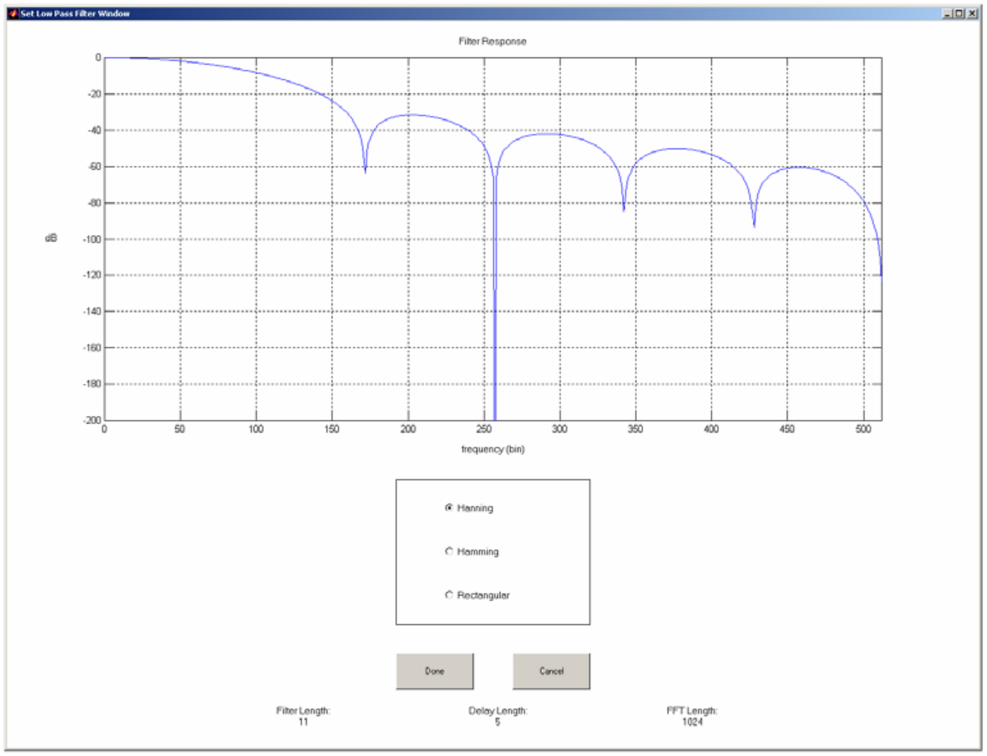


Fig 2.1.8 ABC-Analyzer ‘Time Delay’ Sub-Menu

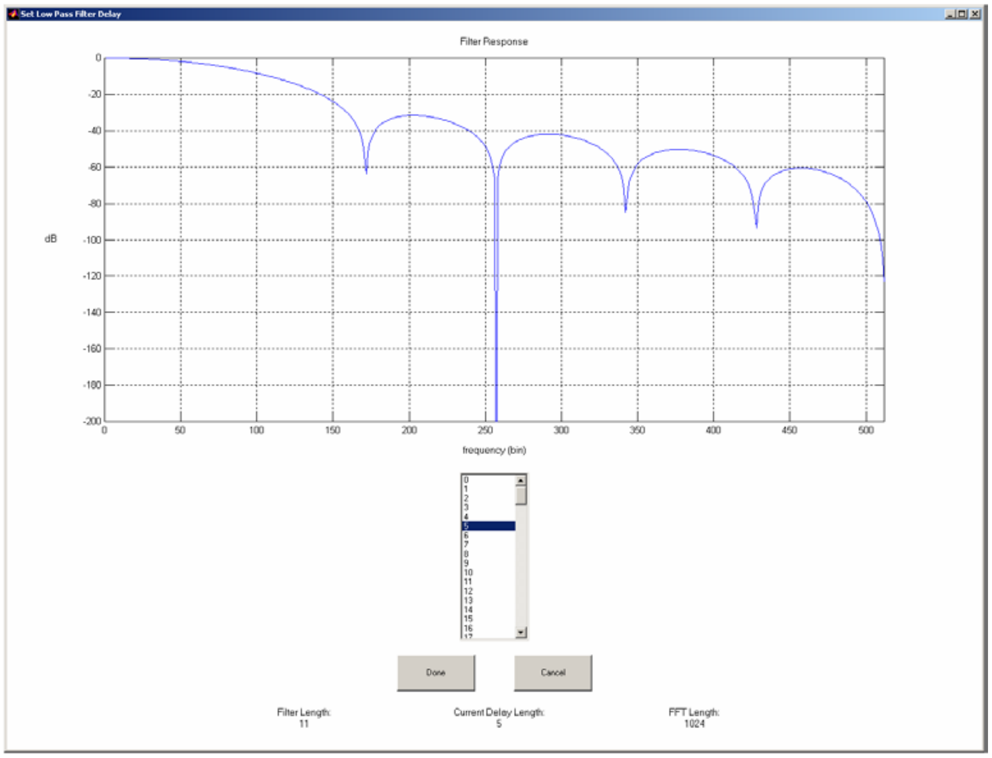
Referring to Figures 2.1.9 and 2.1.10, the ‘Filter Type’ option opens a new window to allow the user to switch that stage’s low pass filter type. It can be switched between ‘Hanning’, ‘Hamming’, and ‘Rectangular’ filters. This parameter can be found in all of the stages except for the last one.



[✓ Hanning] Hamming Rectangular

Filter Length: 11
Delay Length: 5
FFT Length: 1024

Fig 2.1.9 Filter Type Selection Window



Filter Length: 11
Current Delay Length: 5
FFT Length: 1024

Fig 2.1.10 Filter Delay Selection Window

The ‘Filter Delay’ option works similarly to the ‘Filter Type’ option, except that it allows the operator to set the amount of frequency averaging that the selected stage uses. This option is also located in the submenus of all but the last stage. The two windows for the ‘Filter Type’ and ‘Filter Delay’ selections were taken from the previously existing ABC implementation [1].

Next, the ‘Threshold Mode’ option selects the “mode” that applies to the stage one threshold, as shown in Figure 2.1.11. Since this option applies only to the stage one threshold, it is only listed in the submenu of the first stage. The two modes are ‘Float’ (the default mode) and ‘Hold’. The actual implications of these modes will be discussed in the section on the ABC-analyzer display.

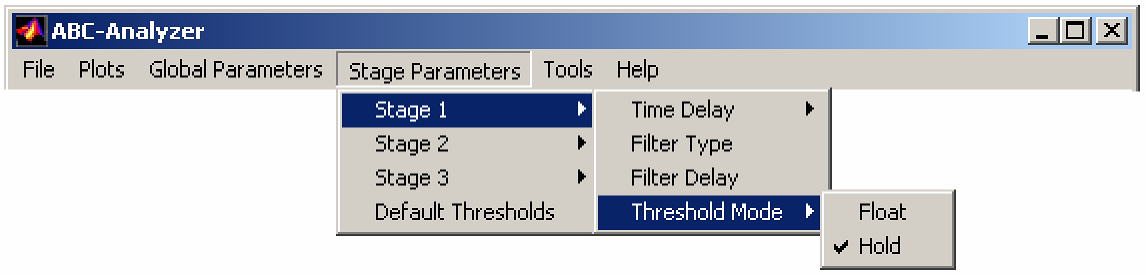


Fig 2.1.11 ABC Analyzer ‘Threshold Mode’ Sub-Menu

Finally, the ‘Default Thresholds’ menu option, located directly under ‘Stage Parameters’, sets the detection threshold for all stages back to the default. For the first stage, it sets the threshold 3 dB above the noise floor estimate. For the other stages, it varies the thresholds between 0 and 6.

Referring to Figure 2.1.12, the 'Tools' menu is the simplest menu in the window. This menu only has one option: 'Show Toolbar'. This option toggles the display of the figure toolbar, so that if it is not needed, it can be hidden to give more room for the display.

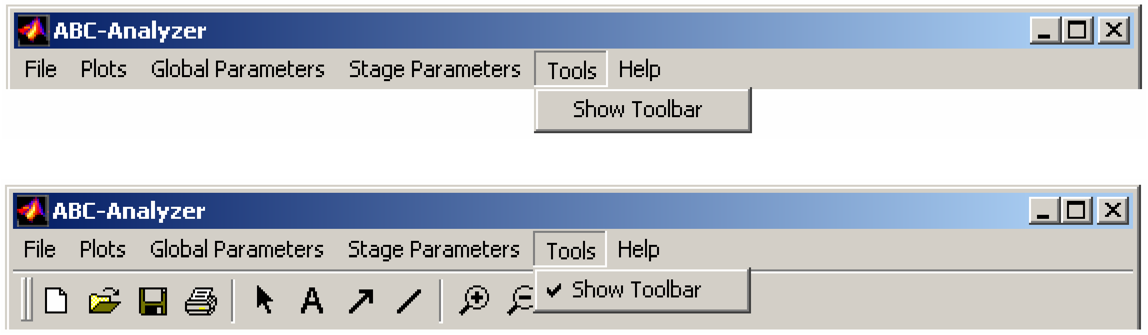


Fig 2.1.12 ABC-Analyzer 'Show Toolbar' Menu

The final menu in the ABC-Analyzer window is the 'Help' menu, as shown in Figure 2.1.13. The options under this menu give simple information on the other menus, which were previously discussed here. It also gives general information on the algorithm and program as a whole.

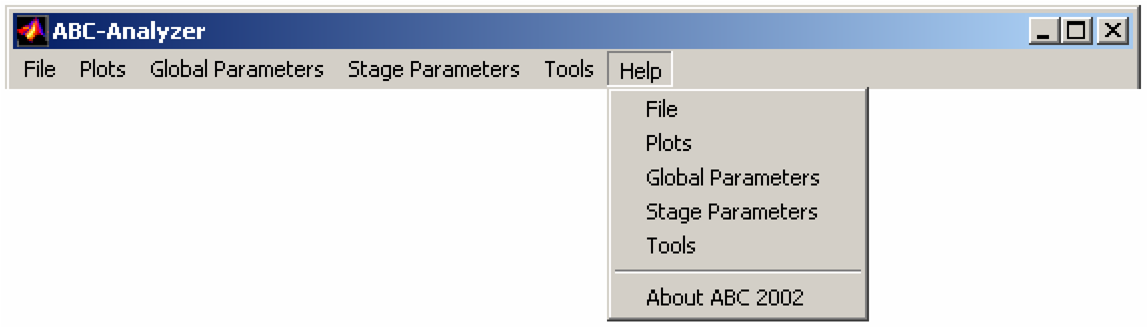


Fig 2.1.13 ABC-Analyzer 'Help' Menu

2.1.2 ABC-Analyzer Display

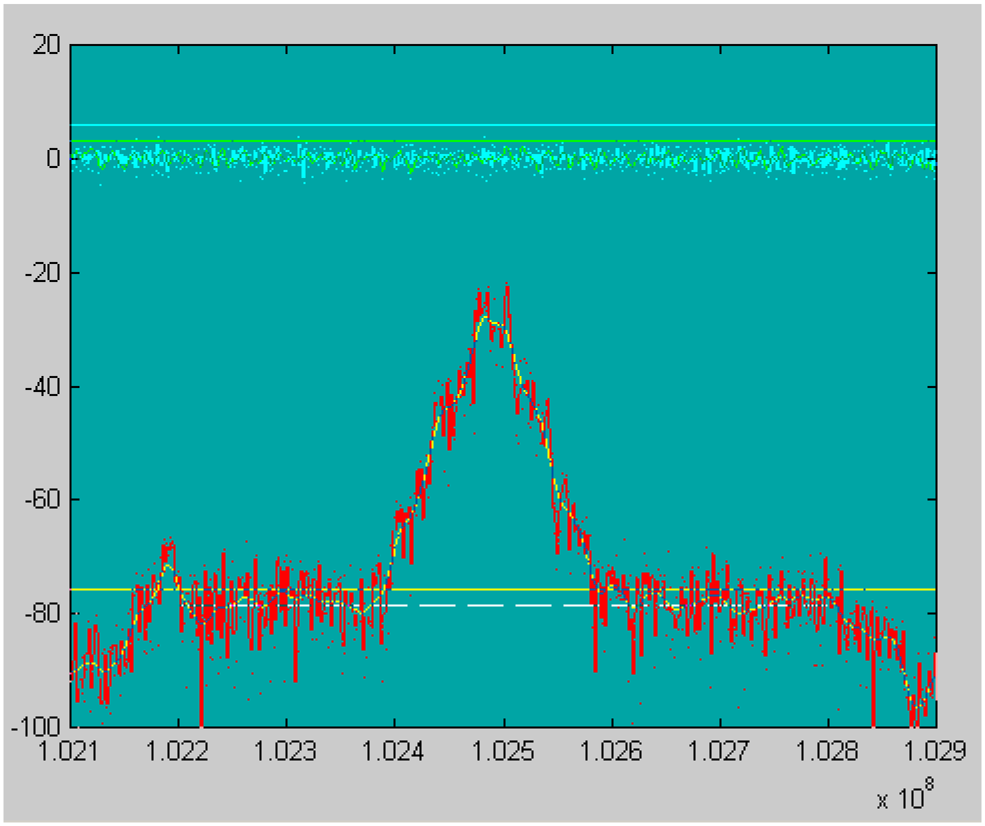
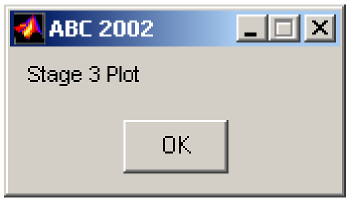


Fig 2.1.14 ABC-Analyzer Display

In the ABC-Analyzer display, the user will see a running spectral display of not only the input signal, but also the spectral display for the output of each stage of the algorithm. This is shown in Figure 2.1.14. The horizontal axis represents frequency in Hz, while the unit of the vertical axis is dBx. Each plot trace is a different color, so that the operator can better distinguish between the stages. Also plotted, are the noise floor estimate and the detection thresholds. The noise floor estimate is plotted as a white, dashed line. Its placement and size are determined by the n.f.e. and filter parameters that are set under the ‘File’ menu. The detection thresholds are plotted as horizontal lines, and are the same color as the stage output plot that they represent. The first stage threshold defaults to the 'Float' mode. This means that instead of staying at one value (as in the 'Hold' mode), it will float a specified distance above the noise floor estimate.

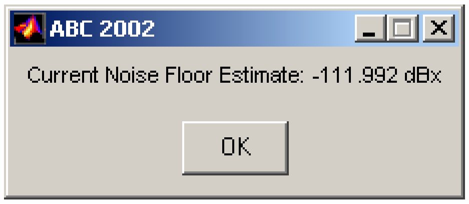
The display section of the ABC-Analyzer window is designed to be just as interactive as the window's menubar. Clicking either the left or right mouse button over any plot trace will tell the operator which stage that trace corresponds to, as seen in Figure 2.1.15.


ABC 2002

Stage 3 Plot

Fig 2.1.15 Plot Identification Pop-Up

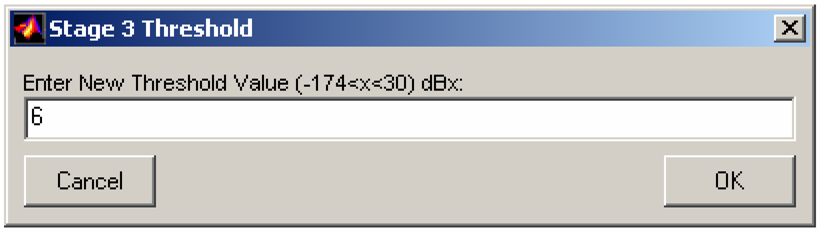
Similarly, clicking either button on the noise floor estimate line will display the current running estimate value as shown in Figure 2.1.16.


ABC 2002

Current Noise Floor Estimate: -111.992 dBx

Fig 2.1.16 Noise Floor Estimate Pop-Up

The detection thresholds are different though. All of the threshold values can be edited either graphically or numerically. Left clicking on any of the thresholds, makes them follow the cursor until they are clicked on a second time. If the stage one threshold is clicked on and it is the 'Float' mode, this process will also automatically switch it over to the 'Hold' mode. Clicking the right mouse button on any of the threshold lines opens a dialog window to allow the thresholds to be edited numerically. See Figure 2.1.17.

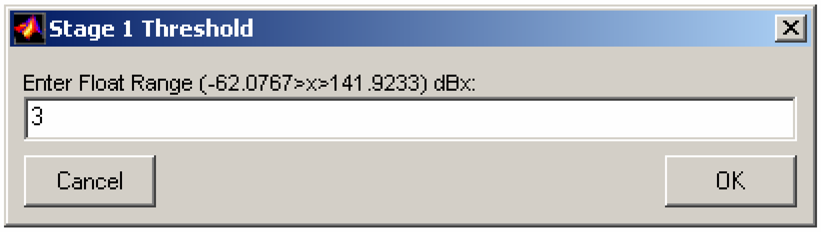


Enter New Threshold Value (-174<x<30) dBx:

6

Fig 2.1.17 Numerical Threshold Editor

The number that appears in the edit box is the current value of that threshold. Referring to Figure 2.1.18, if the stage one threshold is in the 'Float' mode, right clicking on it will allow access to the distance between the threshold and the noise floor estimate.



Enter Float Range (-62.0767>x>141.9233) dBx:

3

Fig 2.1.18 Numerical Threshold Editor for Float Mode

2.2 Signal Object Display Window

The Signal Object Display window is the second major window of the new GUI, and is based on a concept proposed by IFEC/Dr. Noga. Its purpose is to display the running output of the ABC process and the new Signal Object Creation algorithm. This window also has it's own menubar and toolbar, as shown in Figure 2.2.1.

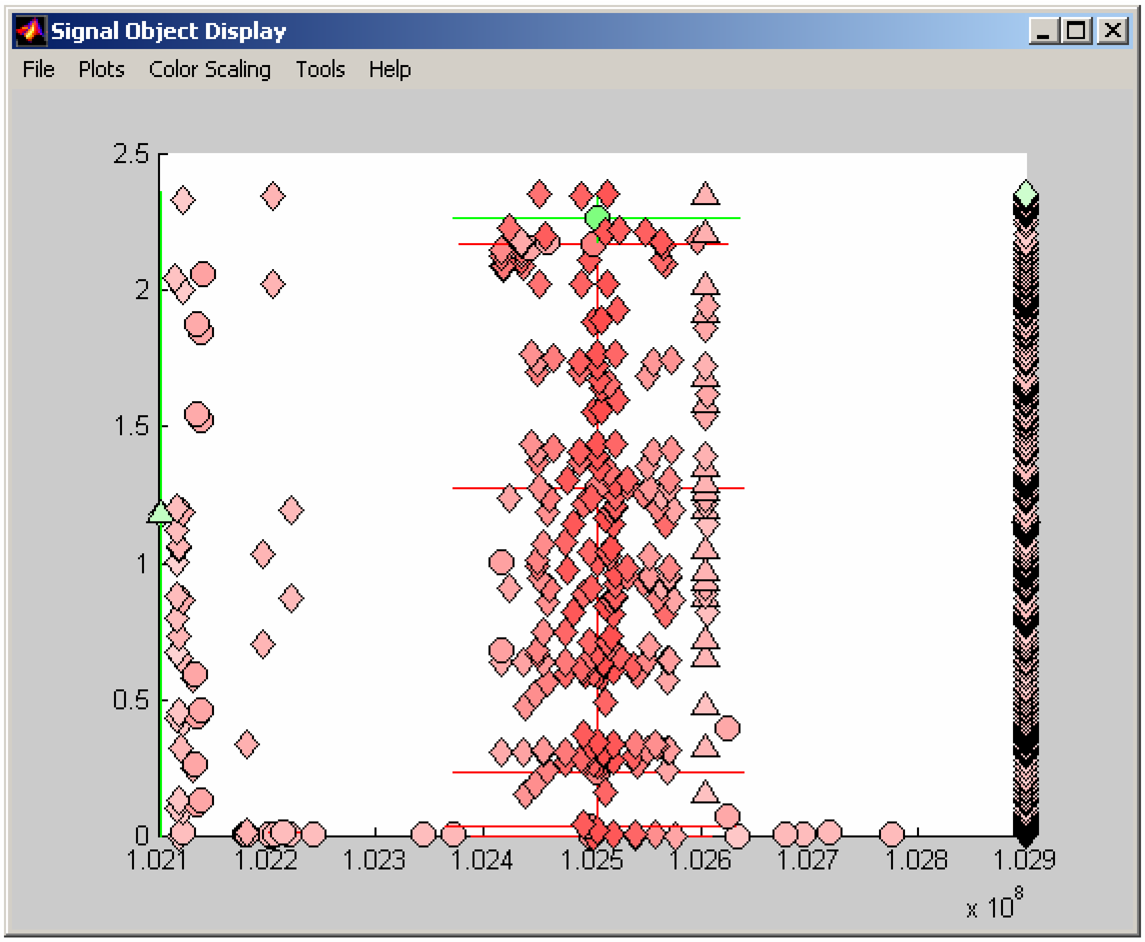


Fig 2.2.1 Screen Shot of the Signal Object Display Window

2.2.1 Signal Object Display Menus

The first menu in the Signal Object Display window is the 'File' menu, in keeping with the standard Windows menubar. See Figure 2.2.2.

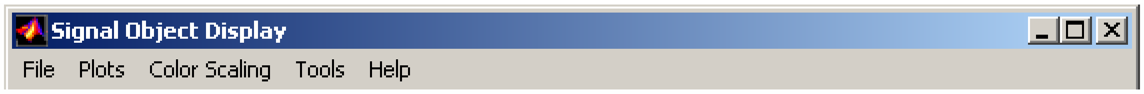


Fig 2.2.2 Signal Object Display Window Menu Bar

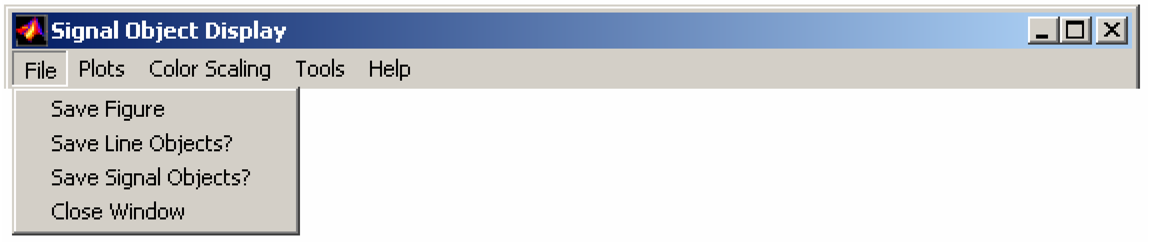


Fig 2.2.3 Signal Object Display 'File' Menu

Referring to Figure 2.2.3, the first option in the 'File' menu is the 'Save Figure' option. This allows the user to save the Signal Object Display as a Matlab .fig file. The next two options tell the program whether or not to write the detected objects to text files. New files will always be created each time the program is run, but these options toggle whether or not the program will append objects into the file. The objects written will be discussed in more detail in paragraph 3.1.1. The last option under this menu is the 'Close Window' option. This not only closes the display window, but also ends execution of the program and closes all other windows as well.

The second menu is the 'Plots' menu, as shown in Figure 2.2.4.

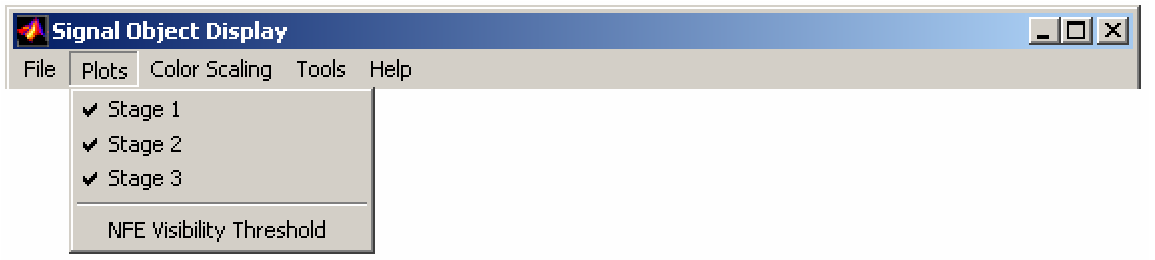


Fig 2.2.4 Signal Object Display 'Plots' Menu

As in the ABC-Analyzer window, this menu contains options that adjust the appearance of the display. This menu includes the ability to toggle the visibility of the outputs from the individual stages. It also contains the 'NFE Visibility Threshold' option. The value edits the visibility of the displayed signal objects based on their (S+N)/N value. Any objects with a ratio value lower than the threshold are made invisible. Conversely, any object that has a higher ratio value is made visible.

The next menu is the 'Color Scaling' menu, as shown in Figure 2.2.5. The options under this menu adjust the scaling method used to set the color intensity of the displayed signal objects.

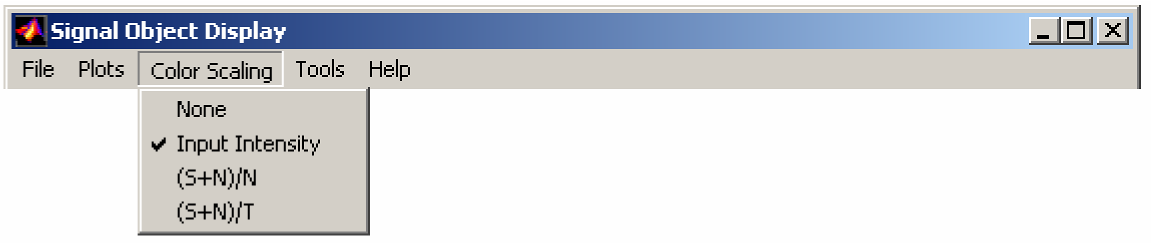


Fig 2.2.5 Signal Object Display 'Color Scaling' Menu

The four options are: 'None'; 'Input Intensity' (based on the intensity of the input signal at the site of the detection); '(S+N)/N' (based on the signal + noise to noise ratio of that detection); and '(S+N)/T' (based on the maximum output to threshold ratio of the stage output over that detection area). When selected, these options apply to all stages, and to all following time segments. It does not change the color scaling of any previously plotted objects, unless those objects are still active ("active" will be defined in the section on the Signal Object Creation Algorithm). The scaling functions are based on stretching the sine curve from $-{}^{\pi}/_{2}$ to $+{}^{\pi}/_{2}$ to over the range of possible values for the selected option. This function was chosen because it gives more separation over the center range, where it is needed the most.

The fourth option in the Signal Object Display window is the 'Tools' menu shown in Figure 2.2.6.

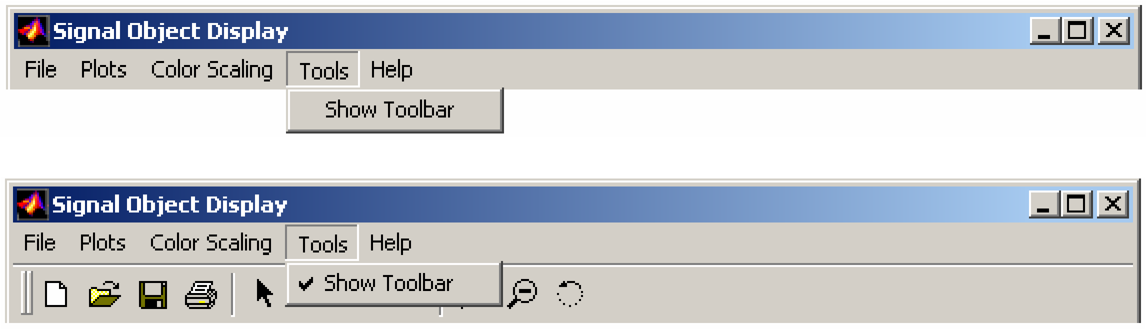


Fig 2.2.6 Signal Object Display 'Tools' Menu

The only option under this menu is again the 'Show Toolbar' option. This works the same way as in the ABC-Analyzer window, in that it simply toggles the visibility of the figure toolbar.

The final menu of this window is the 'Help' menu. As shown in Figure 2.2.7, the options under this menu give brief descriptions of the other menu items in the Signal Object Display window.

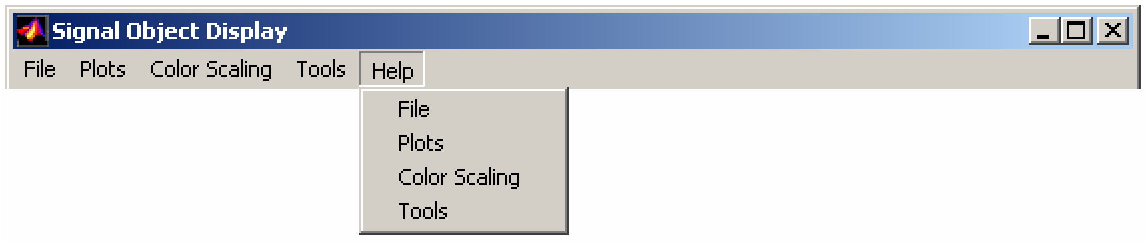


Fig 2.2.7 Signal Object Display 'Help' Menu

2.2.2 Signal Object Display

This display presented in Figure 2.2.8 shows all of the signal objects created by the new Signal Object Creation algorithm. As on the ABC-Analyzer display, the horizontal axis shows frequency in Hz. However, in this window, the vertical axis represents time in seconds. Different shapes indicate objects that were detected in different stages of the ABC process. Circles represent stage one; diamonds represent stage two; triangles, stage three; squares, stage four; and stars represent stage five detections.

If an object was detected in the most recent time segment, it is considered active. The object is then plotted in green. Objects that are no longer being detected are considered expired. Those objects are plotted in red.

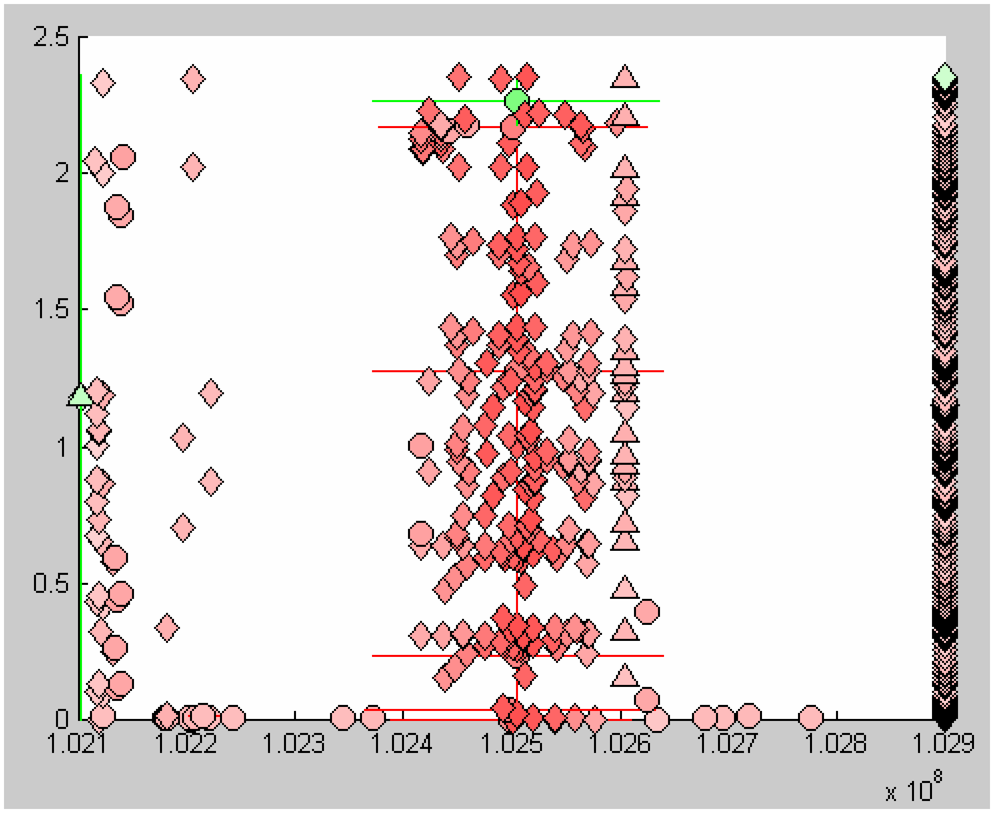
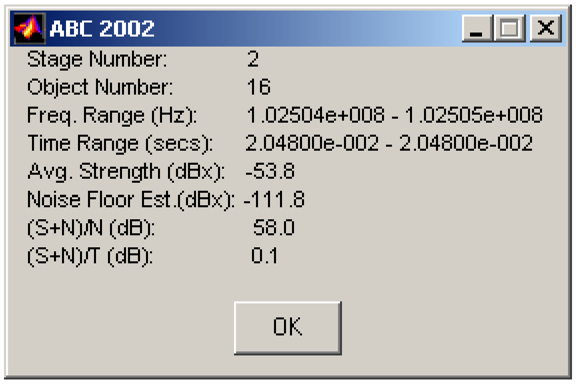


Fig 2.2.8 Signal Object Display

The stage marker shapes are displayed at the center of the object. Coming from that center, are two lines. The vertical line denotes the time period that the object was detected over. The horizontal line denotes the maximum and minimum frequencies that the object was detected at over the duration of the detection. These lines are also plotted in red or green, as it applies to that object.

The color intensity of the plotted symbol varies according to the scaling method used. A weaker color corresponds to a weaker value, whether it is for the (S+N)/T, (S+N)/N, or Input Intensity methods. This scaling applies only to the object symbol, not the time and frequency range lines.

The display in this window is also interactive. Clicking either mouse button on an object will bring up a window that displays various information about that particular signal object, as shown in Figure 2.2.9.



```
Stage Number:           2
Object Number:          16
Freq. Range (Hz):       1.02504e+008 - 1.02505e+008
Time Range (secs):      2.04800e-002 - 2.04800e-002
Avg. Strength (dBx):    -53.8
Noise Floor Est.(dBx):  -111.8
(S+N)/N (dB):            58.0
(S+N)/T (dB):            0.1
```

Fig 2.2.9 Object Information Pop-Up

This information includes the stage number, time and frequency ranges, and average signal strength. The 'Object Number' listed is stage dependant. This is not an overall count of the objects. So, for the case shown, the selected object was the sixteenth object in stage two. The 'Avg. Strength' listing is the average input intensity in the area of this object's detection.

Finally, the '(S+N)/T' value gives the maximum distance between that stage's output and the detection threshold over the course of the selected object's detection.

3.0 Signal Object Creation Algorithm

In an attempt to reduce the size and complexity of the output of the ABC process, an algorithm was created to automatically convert detections into “signal objects”.

3.1 Motivations for a New Algorithm

In the previous ABC implementation, a color-coded binary detection grid was displayed. In the example shown in Figure 3.1.1, the display shows in excess of 100 time segments. In the new implementation, it would be necessary to have thousands of time segments displayed on the screen at once.

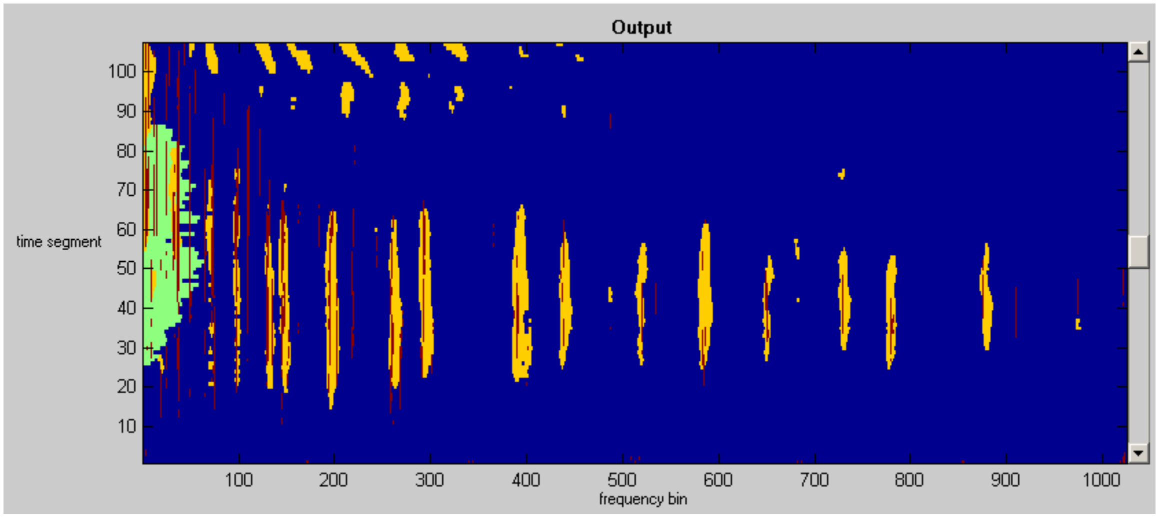


Fig 3.1.1 Previous ABC Output Display

For real-time scenarios, using this previous display would have two problems: more detections than screen pixels, and the scroll speed. With current monitors, resolution usually goes up to 1200x1600 pixels. At a sampling rate of 1.6 MHz, and using an FFT size of 1024 bins, one second of data would take up over 1500 time segments. Even with a full screen display, some detections could be missed. Also, because the screen would have to be refreshed every second, the operator would miss a considerable amount of information. This led to the need for a new output system.

3.2 Constraints

In order to work real time with the ABC algorithm, the signal object creation process would have to work while only given one time segment at a time. Also, the output would have to be plausible. Finally, it would have to be useful to a human operator. This means that it would have to combine the correct detections together, while weeding out noise, and overlapping signals. At the same time, it must also be able to display the new information for a longer time period at higher sampling rates, and in such a way that screen resolution and refresh rates would not result in lost data.

3.3 Algorithm Description

The new algorithm works in two passes. The first pass combines all consecutive detections within the current time segment. The second pass compares these “line objects” in the current segment with the active line objects from the previous time segment. This process occurs independently in each stage, such that each stage has its own list of detected signal objects.

3.3.1 First Pass

In the first pass, the new algorithm uses the binary detection grid for the current time segment from each stage output. It combines all of the frequency-consecutive detections into "line objects", as well as collects information about the signal strength. The list of line objects is actually a Nx8 array, where N is the number of objects. The 8 elements of the array are shown in Table 3.3.1.

Start freq.	End freq.	# active objects it overlaps	# of related signal object	Length of object	Input signal strength	Time detected	ABC stage output strength

Table 3.3.1 Line Object Layout

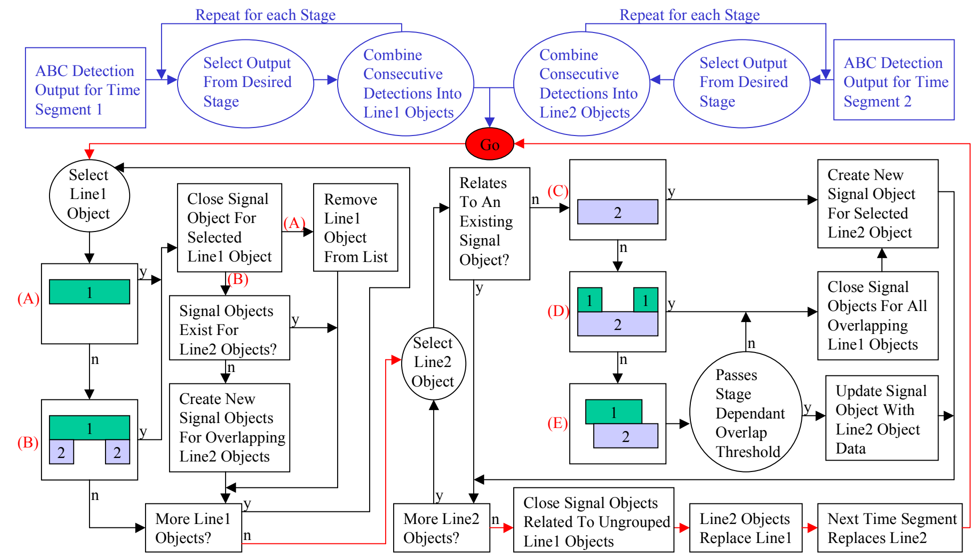
This list is then saved, as it contains all of the active objects. On the first time segment, the second pass is skipped, as there is no previous time segment to compare to. New signal objects are then created for all current line objects, and the algorithm progresses to the next time segment.

In the flowchart shown in Fig. 3.3.3, the blue objects designate the first pass. All other objects denote functions of the second pass. Signal objects are ordered as Nx11 arrays, where N is the number of signal objects for that stage. The 11 elements of the arrays are shown in Table 3.3.2.

Obj #	Min Freq	Max Freq	Start Time	End Time	Avg Signal Power	Stage #	Total Area	ABC Stage Output Strength	$\frac{S+N}{N}$	$\frac{S+N}{T}$

Table 3.3.2 Signal Object Layout

3.3.2 Second Pass



Repeat for each Stage
Repeat for each Stage
ABC Detection Output for Time Segment 1
Select Output From Desired Stage
Combine Consecutive Detections Into Line1 Objects
Combine Consecutive Detections Into Line2 Objects
Select Output From Desired Stage
ABC Detection Output for Time Segment 2
Go
Select Line1 Object
Close Signal Object For Selected Line1 Object
Remove Line1 Object From List
Signal Objects Exist For Line2 Objects?
Create New Signal Objects For Overlapping Line2 Objects
More Line1 Objects?
Relates To An Existing Signal Object?
Select Line2 Object
More Line2 Objects?
Create New Signal Object For Selected Line2 Object
Close Signal Objects For All Overlapping Line1 Objects
Passes Stage Dependant Overlap Threshold
Update Signal Object With Line2 Object Data
Close Signal Objects Related To Ungrouped Line1 Objects
Line2 Objects Replace Line1
Next Time Segment Replaces Line2
(A)
(B)
(C)
(D)
(E)

Fig 3.3.3 Signal Object Creation Algorithm Flowchart

This pass compares overlapping objects from the first pass of the current time segment (line2) and the first pass of the previous time segment (line1). In doing this, the algorithm checks for five specific overlap cases. The algorithm checks for the first two cases while cycling through the line1 objects, and checks for the other three while cycling through the line2 objects. The overlap referred to is frequency domain overlap, and occurs between consecutive time segments.

First, the algorithm cycles through the list of line1objects. This process detects the first two overlap cases, and processes the line objects as necessary.

The first case is when a line1 object does not overlap any line2 objects. The algorithm finds the corresponding signal object, and marks it as expired. It then removes this line object from the line1 list, thus eliminating any needless further processing on it.

The second case involves a line1 object overlapping more than one line2 object. When this happens, the line1 object is treated as a branch point and again the related signal object is marked as expired. The line2 objects are then checked to see if related signal objects have been created that correspond to the significant line2 objects. If not, new signal objects are created for each of the multiple line2 objects that are overlapping with the selected line1 object.

When the algorithm is done cycling through the line1 objects, it begins to cycle through the list of line2 objects. This checks for the last three possible overlap cases. While cycling through, the process checks the selected line2 object to see if a corresponding signal object has been created. If one has, the algorithm skips over this line2 object and proceeds to process the next object in the list.

The third case is similar to the first case. It occurs when a line2 object does not overlap any line1 objects. When this happens, a new signal object is created that relates to the line2 object.

The fourth case occurs when a line2 object overlaps more than one line1 object. The line2 object is now treated as an intersection. The signal objects that correspond to all of the overlapping line1 objects are marked as expired, while a new signal object is created that relates to the line2 object.

The fifth, and final, case is when one line1 object overlaps with one line2 object. The amount of overlap is calculated as the percentage of the longer object that touches the shorter object. This number is then compared to a stage dependent threshold. The thresholds vary evenly between 50% for the first stage and 0% for the last stage. If the amount of overlap is less than or equal to the threshold percentage for that stage, the signal object relating to the line1 object is marked as expired, and a new one is created for the line2 object. Otherwise, The signal object tied to the line1 object is updated using the existing information and the new information from the line2 object.

Once the line comparison is complete, the signal objects corresponding to any unclassified line1 objects are marked as expired. The line1 list then is thrown out and replaced by the line2 list. The algorithm repeats this process for each time segment.

3.3.3 Display

The display of the signal objects is tied directly into the creation algorithm. Since the information about an object can change with each time segment, the display must be able to reflect that change at the same time.

Every time segment is now plotted, and since the symbols used to denote objects are larger than one pixel wide, they will always be visible unless covered by another object. If one object covers another, this can be clarified by simply zooming in on the desired region. All of the shapes and colors displayed in the Signal Object Display are selected based on information contained in the signal object list. If the operator needs more specific information, they can click on the object symbol and directly access most of the information contained in that list.

4.0 Other Output

In addition to displaying the calculated information, this new program also creates a few text files containing other information for use in post-processing.

4.1 Parameter Text File

The first output file contains the parameter list for the ABC process, and is aptly named “params.txt”. Any time a parameter is changed, a new line is added to this file telling what parameter was changed by use of an ‘action’ code. Also, this line lists the new parameter values. Another text file, “Legend.txt”, tells the user what the ‘action’ codes mean.

4.2 Line Object Text File

The next output is a text file that contains a list of detected line objects. This file's name is "LineObjs.txt". These line objects are appended to the file, only if the corresponding menu switch is turned on. The file list contains all of the information contained in the objects as discussed in paragraph 3.3.1.

4.3 Signal Object Text File

The third text file lists the detected signal objects, and it is named "SigObjs.txt". Once again, these files are only written to if the menu switch is turned on. The list in the file contains the signal object information as discussed in paragraph 3.3.1.

5.0 Conclusions

After testing, a few conclusions could be made about the new prototype implementation.

5.1 Speed

It has been determined that the prototype implementation is not capable of actually handling real-time data. However, during testing, the program was clocked at speeds up to 10 kHz. That speed relates to the number of samples the program could handle in one second. When this number is compared to sampling rates of the order of 10 MHz that could be used in some collections, it is easy to see that there is a need for speed enhancements, possibly employing parallel-processing.

5.2 Design Comments

After completing the GUI, it was found to be highly satisfactory. Using pull down menus for parameter adjustment allowed for a maximum display area. If other controls had been used, they would cut into the possible size of the display, and make viewing more difficult. Also, the use of menus kept all of the options in one area, making them simpler and easier to find.

5.3 Other Uses

During the period of time that this program was being developed, it was also used for purposes other than as a program model. The main use was as a test for the data collection system that was in use. This program not only detected collected signals, but in certain cases, it could also indicate problems with the collection such as dropped data. While these problems could be detected easily with this program, they might have otherwise been overlooked.

5.4 Acknowledgements

The author would like to extend his gratitude to Dr. Andrew Noga for his guidance and help in understanding the subject matter at hand. Dr. Noga developed the tasking for the work performed. He would also like to thank Rich Floyd for his assistance in both technical and non-technical matters. Finally, the author would like to express his appreciation for the opportunities given to him by the Summer Engineering Aide Program of the Information Directorate of the Air Force Research Lab.

References

[1] AFRL-IF-RS-TM-1999-6, In House Technical Memorandum, "A Signal Energy Detection Implementation," Timothy M. Hughes, December 1999.